VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Marie Crháková

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ANALÝZA SÍŤOVÉHO PROVOZU NA SÍŤOVÉ KARTĚ FPGA
NETWORK TRAFFIC ANALYSIS ON FPGA NETWORK CARD

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Marie Crháková**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. David Smékal**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Studentka: Marie Crháková ***ID:*** 152417
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Analýza síťového provozu na síťové kartě FPGA

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce je navrhnout vhodné metody pro analýzu síťového provozu a následně je implementovat na síťovou kartu FPGA. Zvolený návrh odsimulujte a následně proveďte základní implementaci na hardwaru. Seznamte se s programovacím jazykem VHDL, FPGA kartami NFB, vývojovým frameworkem NDK a platformou Xilinx Vivado.

DOPORUČENÁ LITERATURA:

[1] WOLF, Wayne. FPGA based system design. New Jersey: Prentice-Hall, 2004, 530 s. ISBN 0-13-142461-0.

[2] PINKER, Jiří, Martin POUPA. Číslicové systémy a jazyk VHDL. 1. vyd. Praha: BEN - technická literatura, 2006, 349 s. ISBN 80-7300-198-5.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 27.5.2019

Vedoucí práce: Ing. David Smékal
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Práce pojednává o síťovém referenčním modelu, protokolové architektuře, následně se zaměřuje na hlavičky konkrétních protokolů IP, TCP a UDP. Dále je rozebráno programovatelné hradlové pole od firmy Xilinx se síťovou kartou od Netcope technologies a jeho funkce definované ve firemním dokumentu NDK. Nakonec byly nastudované informace použity pro vytvoření programu pro čítaní paketů, který byl odsimulován a syntetizován pro programovatelné hradlové pole v prostředí Vivado.

KLÍČOVÁ SLOVA

IP, NDK, paměti, programovatelné hradlové pole, TCP, UDP, VHDL, záhlaví protokolu

ABSTRACT

Bachelor thesis contains description of ISO/OSI and TCP/IP, then is focused on packet headers of the protocols IP, TCP and UDP. After that is analysed the FPGA from the company Xilinx with network interface card from Netcope Technologies and their functions defined by Firmware Developer´s Manual. Finally the studied informations was used for create the program for counting packets, that was simulated and synthesized for field programmable gate array through Vivado.

KEYWORDS

FPGA, IP, memory, TCP, packet header, UDP, VHDL, NDK

CRHÁKOVÁ, Marie. *Analýza síťového provozu na síťové kartě FPGA*. Brno, 2019, 44 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. David Smékal,

Vysázeno pomocí balíčku `thesis` verze 3.03; `http://latex.feec.vutbr.cz`

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Analýza síťového provozu na síťové kartě FPGA“ jsem vypracovala samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autorka uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autorky

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu bakalářské práce panu Ing. Davidu Smékalovi za odborné vedení, podporu, konzultace, obrovskou trpělivost, shovívavost a podnětné návrhy k práci.

Brno ....

podpis autorky

Obsah

Seznam obrázků

Úvod

Analýza sítě je celkem široký pojem, jelikož z přenášených rámců a v nich zapouzdřených paketů se dá zjistit hodně informací. Příkladem širokého výběru analýzy sítě může být například program Wireshark, který prošel mnohaletým vývojem několika pracovníků. Nabízí detailní průzkum přijatých dat s přesným umístěním informací. Výše zmíněný program jsem používala pro nadefinování signálů pro simulaci programu. Zkoušet analyzovat síť pomocí programovatelného hradlového pole se síťovou kartou je výhodné vzhledem k rychlosti zpracování dat.

Práce je zaměřená na teoretický rozbor základních obecných mechanismů, které se uplatňují v síti. Konkrétně jsou v teoretické části rozebrány modely sítě ISO/OSI a TCP/IP s jednotlivými vrstvami. Poté se práce zabývá hlavičkami vrstev, z kterých je čerpáno pro realizaci analýzy síťového provozu. Dále jsou rozebrána hradlová pole a paměti. Dalším bodem je stručná historie vývoje jazyka VHDL a hlavní komponenty. Následně naleznete v textu pojednání o vývojové sadě, která umožňuje následnou realizaci práce.

Praktická část se zaobírá rozborem vhodných a nevhodných způsobů řešení pro programovatelné hradlové pole, konkrétně pak čítání prošlých IPv4, IPv6, UDP a TCP protokolů. Práce je zrealizována v jazyce VHDL. Konkrétně navržena pro FPGA od firmy Xilinx na desce se síťovou kartou od firmy Netcope Technologies. Prostředí, skrz které je možno hradlové pole programovat, se nazývá Vivado. Pomocí Vivada se odsimulovala funkčnost programu a v rámci syntézy, byla zjištěna vhodnost pro programovatelné hradlové pole.

1 Teoretická část studentské práce

1.1 Síťová architektura

Pojem síťová architektura nepopisuje přímo konkrétní systém, ale označuje souhrn pravidel a funkcí. Jedná se o popis struktury sítě, kterou by měl určitý návrh obsahovat. Komunikace skýtá několik složitých dílčích funkcí, pro přehlednost a úplnost ji rozdělujeme do několika menších úkonů, které se odehrávají v různých vrstvách sítě, jež jsou seskupeny v jednom celku, neboli modelu. Členění komunikace do vrstev odpovídá posloupnosti vykonávaných činností, které probíhají během komunikace. Každá vrstva obsahuje konkrétní datovou jednotku, protokol, který zajišťuje práci s daty pro danou vrstvu. Pokud protokol nepracuje správně, můžeme ho vyměnit a nemusíme měnit celou architekturu. Funkce protokolů jsou definovány metodami, které mimo jiné zajišťují jejich koordinaci, aby byla garantována funkčnost celého systému.[15]

1.1.1 Sedmi vrstvový síťový referenční model ISO/OSI

Otevřené systémy architektury (OSA) bylo potřeba normalizovat, aby všechna koncová zařízení vyhovující normám mohla být připojena do stejné sítě. Vznikl referenční model OSI (z anglického Open System Interconnection) s označením IS 7498 roku 1984, který vytvořila mezinárodní normalizační organice ISO (angl. International Organisation for Standardization). Problémem se začala zabývat i Mezinárodní telekomunikační unie (International Telecommunication Union ITU-T, původně se nazývající CCITT), která model OSI přijala a vydala pod označením X.200. Norma obsahuje pojmy reálný systém, což je kterékoli zařízení schopné zpracovávat a přenášet informace (např. počítač, terminál, periferní zařízení...), přičemž jako otevřený reálný systém označujeme všechna výše uvedená zařízení splňující normu OSI. Základní forma normy obsahuje 7 vrstev, avšak nespecifikuje práci protokolů, které se věnují až navazující opatření a doporučení Mezinárodní telekomunikační unie. [15]

Každá ze sedmi vrstev, až na fyzickou, využívá služeb nižší vrstvy a svoje služby poskytuje sousední vyšší vrstvě. Spolupráce sousedních vrstev probíhá přes přístupové body služby (SAP – Service Access Point). Vrstvy mohou komunikovat i s různými entitami v rámci jedné vrstvy. Při průchodu dat od vyšší vrstvy k nižší se protokolová datová jednotka stále zvětšuje o záhlaví vrstev, dochází k tzv. zapouzdření informace. Při opačném směru postupu, tedy od fyzické vrstvy po aplikační, dochází k opačnému procesu, rozbalení datové jednotky a odstranění přidaných záhlaví jednotlivých vrstvových protokolů. [15]

1. Fyzická vrstva

Ve fyzické vrstvě se vytváří datový okruh, prostřednictvím kterého dochází k přenosu bitů nebo značek po přenosovém médiu (kabel či vlna). Datová cesta má podobu plného nebo poloviční duplexu. Pojem poloviční duplex znamená, že stanice je buď vysílač nebo přijímač. Jeden kanál slouží pouze pro jeden směr komunikace. Plný duplex označuje vysílání, které probíhá v jednom kanálu v obou směrech. Spojení známe dvoubodové, či mnohabodové. [15] Transportované signály jsou reprezentovány analogově, nebo digitálně, základně, nebo širokopásmově. Jsou přenášeny synchronně, či asynchronně a rozděleny na určité kanály.

Služba na fyzické vrstvě zajišťuje zahajování a ukončování fyzického spojení. Seřazuje bity, značky, oktety. Podává hlášení o chybách vyšší spojové vrstvě. Médiem putujícím sítí může být elektřina (kabel), světlo (optické vlákno), rádiové, infračervené nebo laserové vlny. Charakterizují ji změny napětí a jejich časování, rychlost přenosu, maximální vzdálenost pro přenos, fyzické přípojky k přenosovému médiu (konektory), fyzická topologie datové cesty. Zařízení, jenž v ní pracují se signalizací, se jmenují např. transceiver na síťové kartě, opakovače, rozbočovače, konektory. [11]

2. Spojová vrstva

Spojová vrstva umožňuje zahajování, udržování a ukončení vytvořených spojení v rámci fyzických okruhů na fyzické vrstvě. Dále zajišťuje formátování rámců, synchronizaci, řadí je, řídí jejich tok. Detekuje a opravuje chyby, neopravitelné hlásí síťové vrstvě. Zjišťuje, vyměňuje a dodržuje parametry výkonnosti svých služeb. Síťové vrstvě umožňuje řízení propojování datových okruhů na úrovni fyzické vrstvy. Jednou z jejich úkolů je fyzická adresace. [15]

3. Síťová vrstva

Pomocí síťových (logických/IP) adres zprostředkovává komunikaci v síti, směrování, vybírá nejvhodnější cestu a přenáší datové jednotky označované jako pakety či datagramy i přes různé podsítě/sítě. Datové jednotky si formátuje do paketů. Proces směrování může provádět díky mapování logických i fyzických adres, vybírá vhodnou cestu přes další otevřené systémy. Uskutečňuje komunikaci využitím služeb nižší spojové vrstvy.

Poskytuje další služby, vybírá přenos a jeho kvalitu, síťově adresuje, zahajuje, udržuje a ukončuje síťové spojení, identifikuje koncové body. Detekuje, oznamuje, opravuje vzniklé chyby. Stará se o pořadí paketů, řídí datový tok, uvádí síťová spojení opět do původního stavu, přijímá potvrzení. Rozlišujeme dva typy síťové služby: se spojením a bez spojení (tzv. datagramová). První jmenovaná vytvoří, udržuje a

ukončuje propojení, směrovač je zatížen sledováním přenosu na této vrstvě. Naopak datagramová služba zasílá různé datagramy jedné zprávy nezávisle na sobě. [15]

4. Transportní vrstva

Transportní vrstva je prostředníkem mezi vyššími vrstvami, které se zabývají aplikacemi a nižšími vrstvami, které zajišťují přenos. Je realizována pomocí doplňkového softwaru, tedy síťový klient v koncových zařízení, který přijímá tok dat od aplikace a rozděluje je do menších PDU segmentů či datagramů. Každá z těchto jednotek obsahuje záhlaví, které obsahuje rozpoznání cílové a koncové aplikace, pro kterou jsou data určena. U TCP/IP jsou aplikace na transportní vrstvě rozlišeny zdrojovými a cílovými čísly portů. Všechny nižší protokoly pod transportní vrstvou se označují jako koncové, jelikož zprostředkovávají komunikaci koncových otevřených systémů. [15]

Transportní vrstva nesměruje, ale poskytuje vyšší relační vrstvě službu s či bez spojení. Přičemž zmíněná služba se spojením logicky navazuje, udržuje a ukončuje spojení. Při navazování spojení transportní vrstva pracuje na získání síťového spojení, které bude vyhovovat relační vrstvě z hlediska ceny a kvality. Rozhodne o vhodnosti mnohonásobného sdružení, určí ideální délku protokolové datové jednotky tedy segmentu, přiřazuje síťovou adresu transportní adrese, zjišťuje možnosti ohledně transportních spojení. Při přenosu dat vybírá velikost fragmentů, uspořádává je, řetězí, několikanásobně sdružuje (tzv. multiplexuje) nebo rozvětvuje identifikované spojení, řídí tok, detekuje chyby a opravuje je. Pro ukončení oznámí důvod a rozpozná konkrétní transportní spojení. [15]

Transportní vrstva přiřazuje adresy příslušné dané vrstvě. Mezi dvojicí adres může vytvořit více spojení, které jsou limitovány pouze kapacitou prostředků. Jeden transportní úkon může vyřešit více relačních spojení. Několik transportních adres může patřit k jedné síťové. Kvalita služeb je závislá na třídě služeb, kterou vyžaduje relační vrstva, jenž je dodržena po celou dobu transportu. Třídy obsahují parametry jako propustnost, doba přenosu, zbytková chybovost atp. [15]

5. Relační vrstva

Relační vrstva řídí interakci mezi kooperujícími prezentačními entitami, navazuje a ukončuje relační spojení, obsluhuje různé typy přenosu zpráv, stará se o synchronizaci. Spolupracuje s transportní vrstvou, pomocí ní řídí tok, avšak spojení (mezi transportní a relační vrstvou) můžou mít různou délku trvání, takže několik transportních spojení je přiřazeno k jednomu relačnímu a naopak. [15]

6. Prezentační vrstva

Prezentační vrstva zajišťuje jednotnou strukturu zpráv (dat, či datových struktur), transformuje syntax zprávy podle potřeby pro vysílající, přijímající entitu, či pro přenos. Přenosová syntaxe se vyjedná mezi komunikujícími prezentačními entitami. Případně se i zabývá (de)kompresí a (de)šifrováním dat. Funkcí vrstvy je požádat o vytvoření a zrušení relace. Jako jediná vrstva v modelu může určitým způsobem zprávy modifikovat. Formátování není potřeba vždycky. Změna struktury zprávy je potřeba například pro tzv. terminálovou emulaci, příkazy z terminálu (znaky ASCII z PC) musí převést na formát srozumitelný pro centrální počítač(formát EBCDIC). [15]

7. Aplikační vrstva

Úkolem aplikační vrstvy je poskytnutí přístupu ke komunikačnímu systému a možnosti spolupráce. Přenáší zprávy, zjišťuje stupeň připravenosti komunikující entity, dále parametry komunikace a možnost jejich uskutečnění. Dává pověření pro přenos, vyjednává mechanismy ochrany, zodpovídá za opravu chyb a celistvost dat, rozhoduje o přiměřenosti prostředků. Dohodne se na tarifu a kvalitě služby a omezeních syntaxe zpráv. Synchronizuje aplikace, vybírá podobu dialogu zahrnující postup zahájení a zakončení. Pro poskytování služeb využívá funkce, které ji zajišťují nižší vrstvy. Je unikátní, jelikož funkce v této vrstvě mohou provádět i lidé. [15]

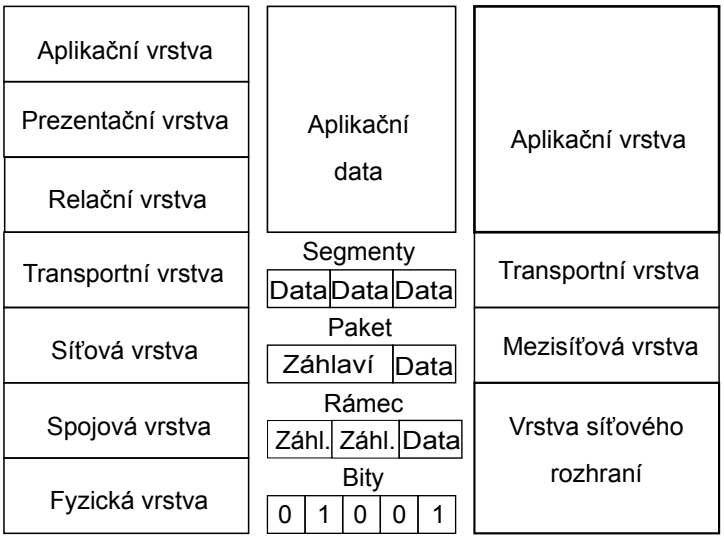
1.1.2 Protokolové architektury TCP/IP

Jeden z nejvyužívanějších protokolů je soubor protokolů Transmission Control Protocol/Internet protocol (TCP/IP), veškeré informace o Internetu jsou publikovány v RFC – Request for Comments, což je norma vytvořená skupinou IETF – Internet Engineering Task Force. Sedmivrstvý model OSI je sloučen do čtyř vrstev viz 1.1. První vrstva síťového rozhraní odpovídá dvěma nejnižším vrstvám fyzické a spojové. Druhá vrstva mezisíťová (angl. internet layer) má stejné funkce jako síťová v OSI, stejně jako třetí transportní vrstva se shoduje i jménem s vrstvou v OSI modelu. Poslední aplikační vrstva zahrnuje tři nejvyšší vrstvy modelu OSI, konkrétně relační, prezentační a aplikační. [15]

Vrstva síťového rozhraní

Optické vlákno

Optické vlákna bývají složené ze dvou vrstev skla, jeden typ pro jádro a jiný pro obal, paprsek se odráží mezi dvěma druhy skla. V novějších kabelech je paprskem většinou laser. Existují jednovidová, či vícevidová vlákna. Vícevidová vlákna se ještě



Obr. 1.1: Porovnání síťových architektur OSI a TCP/IP [1]

dělí na vlákna se skokovou změnou indexu lomu a na s gradientním průběhem indexu lomu (elipsovitý odraz). První zmíněný vícevidový typ funguje na principu totálního odrazu, kdy index lomu vnitřního prostředí musí nabývat vyšší hodnoty, než index lomu vnějšího prostředí, od kterého se paprsek odráží. [2]

Pro přenos médiem se volí paprsek s nízkým optickým odporem typicky 850 nm, 1300 nm či 1500 nm. Optickým rozhraním je simplex, takže na jedné straně se nachází vysílač a na druhé přijímač. Pro oboustranný provoz je zapotřebí dvou vláken. [5]

Typ vlákna	**Průměr jádro/plášť**
jednovidové	5–10/125
vícevidové	50/125 a 62,5/125

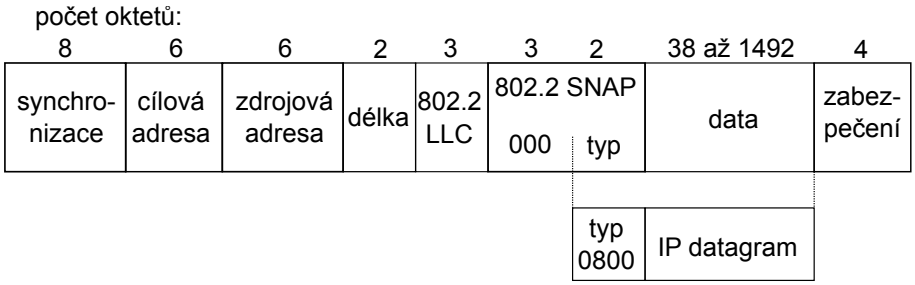
Standardní průměr optických vláken [2]

Ethernet 100 Gb/s

Po výzkumu a testování byl vyvinut standard Ethernet 100 Gb/s IEEE 802.3 ze standardu 10 Gb/s, bylo to možné díky optickým transportním sítím a nápadu vyzkoušet paralelní přenos. Protokol funguje na bázi 10 paralelních linek vycházejících ze standardu 10 Gb/s nebo 4 paralelních vlnových délek s rychlostí 25 Gb/s na frekvenční mřížce 800 GHz o vlnových délkách kolem 1300 nm. Pro převod mezi dvěma technologiemi je zapotřebí využití multiplexování (10:4) a podkanál pro odstranění zkreslení. [12] Jakožto vyšší standard metody Ethernet od 1 Gb se již nevyužívá metoda mnohonásobného přístupu prostřednictvím naslouchání nosné s detekcí kolizí CSMA/CD, ale poloviční nebo plného duplexu.

Rámec Ethernet standard IEEE 802.3 [15]

- **synchronizace – preambule** – obsahuje 10101... poslední oktet 10101011, který se nazývá omezovač počátku rámce (SFD – Starting Frame Delimiter)
- **zdrojová a cílová adresa** – fyzické MAC adresy, které jsou dlouhé 16 nebo 48 bitů zdrojová je vždy individuální, cílová může být všech 3 typů
- **délka** – rámce 64 – 1518 bytů, na záhlaví připadá 18 oktetů
- **typ protokolu** – u standardu Ethernet 2 místo pole délka, ale u námi uvedeného standardu ho najdeme v poli SNAP viz 1.2. Pro protokol IPv4 nabývá hodnoty 0800, pro IPv6 8100.
- **data** – data potom jsou dlouhá 46-1500 bytů, přičemž jsou v nich zahrnuta pole LLC a SNAP
- **zabezpečení** – kontrolní součet, vypočítán včetně omezovače počátku, avšak bez zbytku preambule



Obr. 1.2: Rámec Ethernet standard IEEE 802.3 [15]

Síťová karta

Síťová karta (NIC - network interface card) pracuje na dvou nejnižších vrstvách spojové a fyzické. Od síťové vrstvy je paket zapouzdřen do rámce, poté ho předává fyzické vrstvě s odpovídající fyzickou adresou. Bity jsou přenášeny optickým, rádiovým nebo elektrickým signálem. [15]

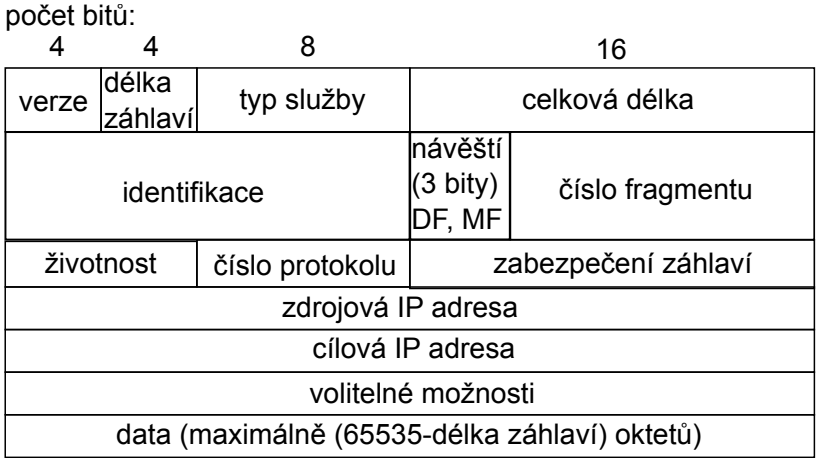
Mezisíťová vrstva

Záhlaví datagramu Ipv4

Minimální délka záhlaví IPv4 je 20 oktetů, maximálně dosahuje délky 60 oktetů [15].

- **verze** – verze protokolu(4)
- **délka záhlaví IPv4** – vyjádřená v 4 bytových blocích, či 32 bitových slovech např. pro hodnotu 4 je záhlaví dlouhé $4 \times 4 = 16$ bytů, nebo $4 \times 32 = 128$ bitů
- **typ služby** – definuje parametry pro zpracování vyšší vrstvou: prioritu, zpoždění, propustnost, spolehlivost, cena

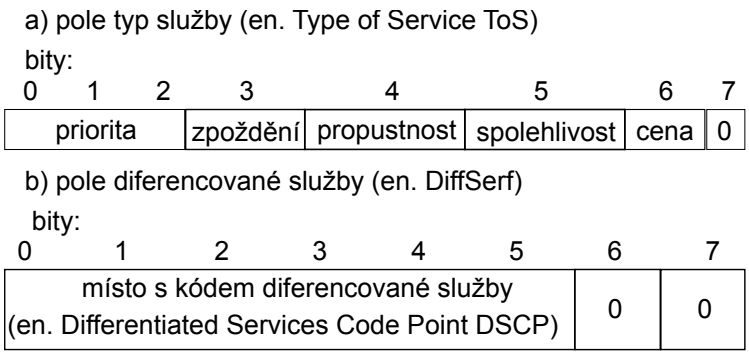
- **celková délka** – vždy násobkem 32 bitů, někdy je třeba doplnit bity pro zarovnání
- **identifikace** – čísluje datagramy, zvyšuje se o jedna s každým přibývajícím datagramem, důležité pro fragmentaci
- **návěští** – je dlouhé 3 bity, přičemž pokud bit 1 DF značí, jestli se má datagram fragmentovat 1 značí ne, 0 ano, bit 2 MF oznamuje, zda je fragment poslední 1 ne, 0 ano
- **číslo fragmentu** – je dlouhé 13 bitů, pořadí fragmentu je definováno počtem dat od prvního fragmentu k aktuálnímu v násobcích 64 bitů



Obr. 1.3: Datagram IPv4 [15]

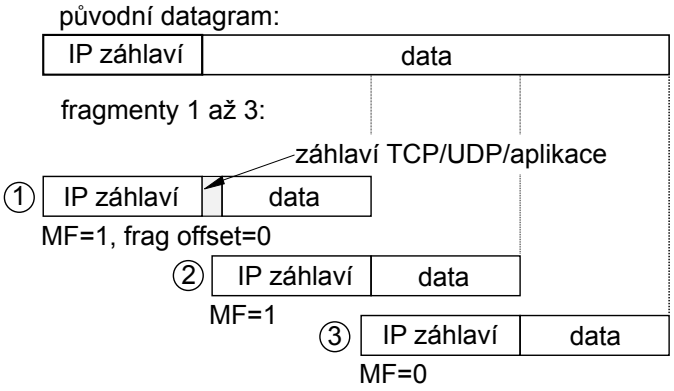
- **životnost** – maximální počet skoků v síti, který se zmenšuje při průchodu směrovačem, zamezuje nekonečnému bloudění paketů, pokud dosáhne hodnoty 0, je zničen a protokol řídících hlášení vyšle zprávu, že čas vypršel
- **číslo protokolu** – udává kód pro protokoly vyšších vrstev (transportní a směrovací protokoly a ICMP – Internet Control Message Protocol), kterým má být zpracován (pro TCP 6, pro UDP 17)
- **zabezpečení záhlaví** – kontrolní součet záhlaví
- **zdrojová adresa** – adresa zdrojového portu o 32 bitech (pouze individuální adresa)
- **cílová adresa** – adresa cílového portu o 32 bitech (individuální, skupinová, nebo všeobecná adresa)
- **volitelné možnosti** – např. záznam cesty sítí, průchod směrovači
- **data** – informace vyšších vrstev (transportní, síťové ICMP/IGMP – Internet Group Managment Protokol, směrovací protokoly)

Typ služby ToS 1.4 zajišťuje kvalita služby QoS, které obsahuje diferencované služby. Pokud má datagram vyšší hodnotu, bude zpracován přednostně [15].



Obr. 1.4: Pole typ služby popis a umístění [15]

V případě, že je povolena fragmentace viz 1.5, provádějí ji směrovače. V koncovém zařízení sítě se segmenty opět seskládají do původní podoby. Minimální délka dat, které musí umět uzly sítě TCP/IP zpracovat, je 576 oktetů. [15]



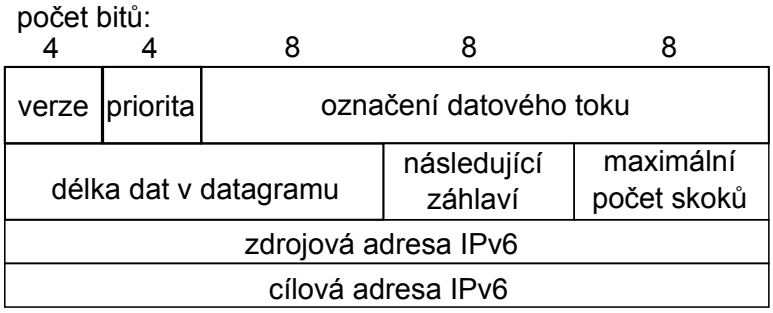
Obr. 1.5: Fragmentace datagramu [15]

Ipv6

Kromě základního záhlaví, které má pevně stanovenou délku, může obsahovat IPv6 ještě několik volitelných záhlaví, které mají sice definované pořadí, ale některá nemusí vůbec obsahovat. UDP by bylo až třetí v pořadí po hlavní hlavičce. [4]

- **verze** – stejně jako u předchozího, rozlišuje verzi protokolu, takže obsahuje číslo 6
- **priorita** – 4/ dle RFC 2460 8 bitů, specifikuje pořadí v rámci sítě, do jaké míry bude upřednostňován
- **označení datového toku** – s jakým výkonem budou směrovače datagram přenášet
- **délka dat v datagramu** – bez základní hlavičky, avšak součet obsahuje i volitelná záhlaví

- **následující záhlaví** – specifikuje dle stejných čísel, jako v IPv4 číslo protokolu
- **maximální počet směrovačů** – při průchodu směrovačem se hodnota sníží o 1, pokud dosáhne hodnoty 0, je zahozen. ICMP vyšle zprávu o dané akci odesílateli.
- **zdrojová a cílová adresa** – 128 bitů, jsou čtyřikrát větší, než u IPv4



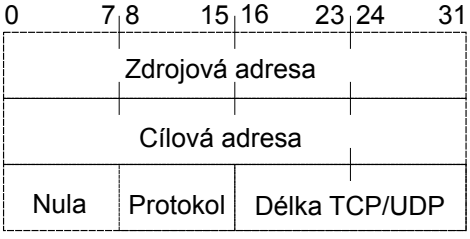
Obr. 1.6: Povinné záhlaví IPv6 [15]

Transportní vrstva

TCP - Transmission Control Protocol

Spolehlivá transportní služba, která je poskytovaná se spojením, jenž musí být před přenosem dat navázáno. Vyžaduje plně duplexní spojení, jelikož potřebuje k započetí odesílání potvrzení od příjemce. Záhlaví obsahuje velikostně proměnlivé tzv. klouzavé okno, které optimalizuje svou velikost. Velikost okna definuje počet segmentů, které se odešlou. Příjemce zasílá postupně potvrzení, v nichž zároveň definuje počet oktetů, který je schopný přijmout. [15]

Tzv. Pseudozáhlaví slouží k zabezpečení TCP i UDP, obsahuje nejzákladnější informace ze segmentu/datagramu. 1.7 Jde jen o fiktivní jednotku, která slouží k výpočtu kontrolního součtu obou protokolů. [15]



Obr. 1.7: Pseudozáhlaví TCP/UDP [14]

UDP – User Datagram Protocol

UDP je nespolehlivá transportní služba vhodná pro aplikace, pro něž může být navazování spojení zbytečně zdlouhavým procesem. Ihned se může začít odesílat a už obsahuje uživatelská data. Oproti TCP má jednodušší hlavičku, jelikož neobsahuje velikost okna, čísel segmentů a potvrzení. 1.8 Daný protokol podporuje odesílání na všeobecnou adresu (255.255.255.255) a na skupinové adresy. Je používán aplikačními protokoly např. SNMP, DNS. Zdrojový port určuje vysílající aplikační proces. Cílový port určuje cílovou aplikační adresu. Délka datagramu bývá v násobcích 32 bitů. [15]

Obr. 1.8: Záhlaví UDP [14]

1.2 Programovatelná hradlová pole

Programovatelná hradlová pole (FPGA, Field Programmable Gate Array) jsou polovodičové vysoce integrovatelné obvody (VLSI – Very large-scale integration), kdy na malém prostoru se nachází co nejvíce elektrických součástek [3]. Obsahují logické programovatelné bloky, které jsou spojené v jeden celek.

Integrované obvody, které jsou vyrobené na zakázku (ASIC, Application Specific Integrated Circuits), jsou optimalizované na funkce, které budou vykonávat [17]. Pracují rychleji a efektivněji díky konkrétnímu využitých součástek. Výhody programovatelných hradlových polí spočívají v rychlosti implementace, jelikož nečekáme na konkrétně navržený pracovní čip pro náš program, můžeme ihned testovat na univerzální topologii. Máme možnost případné opravy chyb, které jsme zjistili při běhu programu. Stejný čip může být použitý na několik různých návrhů.

Programovatelná hradlová pole jsou složená z tranzistorů N a P typu MOSFET, na které můžeme pohlížet jako na spínače. Tvoří je kombinační logické bloky či logické elementy. I/O bloky mohou být naprogramovány jako vstupy, či výstupy, které nabízejí nízký energetický odběr, či vysokou rychlost. Hodinový signál je generován zvlášť, je mu vyhrazena speciální propojovací síť. [19]

1.2.1 Paměti s možností čtení a zápisu

Statická paměť – SRAM

U statické paměti tzv. SRAM se k zapamatování používá kladná zpětná vazba bistabilního klopného obvodu, informace v obvodu zůstane a může být dále využívána jinými obvody, až do přepsání, či vypnutí napájení [13]. Základní stavební jednotkou této paměti jsou tzv. kombinační logické bloky (CLB – combinational logic block), které můžou být také nazývány logické elementy (LE – logic element) [19].

Statickou pamětí je i tzv. lookup table (LUT), která implementuje pravdivostní tabulku. Ze vstupů se vypočítá a uloží pravdivostní tabulka, která je uložena v příslušné adrese patřící ke vstupům. Uvažujeme-li n vstupů, patří k nim 2^n umístění. Základní SRAM nezávisí na hodinovém signálu, funguje jako logické hradlo, takže pokud se změní vstup, po určitém zpoždění se změní výstup. Avšak na rozdíl od hradla, hodnoty můžou být změněny uložením jiných hodnot bitů do SRAM, potom n vstupů je reprezentováno 2^{2n} funkcí (některé z nich vzájemně permutují). Typický logický element mívá od 4 do 6 vstupů, obvykle obsahuje registry řízené hranou (flip-flop) nebo úrovní (latch) a kombinační logiku. [19]

Vrstva obsahuje dvě logické buňky, jedna logická buňka je složena ze dvou čtyřbitových vyhledávacích tabulek. Každá z tabulek může být použita buď jako 16-ti bitová synchronní RAM, nebo jako 16-ti bitový posuvný registr. Každá vrstva obsahuje multiplexor, který se používá ke kombinaci dvou výsledků, které vznikly ze dvou funkčních generátorů vrstvy. Jiné zase zpracovávají výsledek ze dvou vrstev, z čehož pak vzniká výsledek pro celý kombinační logický blok. [19]

Dynamická paměť – DRAM

Dynamická paměť DRAM používá náboj na kondenzátoru, který se nabíjí, či vybíjí přes spínač MOSFET. Avšak náboj je třeba obnovovat, jelikož se po nabití postupně vybíjí. Čtení probíhá skrze zesilovač jako klopný obvod, který porovnává napětí na kondenzátoru s referenčním napětím z čehož vyvodí hodnotu 0/1, jde o jednobitovou paměť. Pro manipulaci s pamětí je využíván řadič, jenž obsahuje čítač adres, multiplexor a řídící obvody. [13]

Synchronní dynamická paměť – SDRAM

Synchronní dynamické paměti vycházejí z předchozího modelu dynamických pamětí, avšak navíc obsahují další obvody, které zrychlují čtení a zápis, usnadňují časování signálu v rámci paměti. Jsou řízeny náběžnou hranou synchronizačního hodinového pulsu, jenž zajišťuje zapsání do registrů. [13]

Rychlejší synchronní dynamická paměť – DDR

Synchronní dynamická paměť DDR vychází opět z předchozí uvedené SDRAM, ale je dvakrát rychlejší, což napovídá již zkratka (DDR – Double Data Rate). Čtení a zápis do paměti je řízen náběžnou i sestupnou hranou. [13]

1.2.2 Permanentní paměti

Uložená data permanentních pamětí nejsou závislé na napájení, zůstávají i po jeho vypnutí. [13]

Pamět pouze pro čtení (ROM – Read only memory)

Obsah je definován již při výrobě a jeho změna není možná. Do paměti se ukládají data pomocí tranzistoru MOS, kde vznikne, nebo nikoli, indukovaný kanál, což vyznačuje dvě úrovně. Proleptá se oxid na stanovených místech. [13]

Programovatelná čtecí paměť (PROM – Programmable ROM)

Obsah může zapsat uživatel, ale obsah je nesmazatelný. K uložení se využívá tzv. plovoucí hradlo, kdy mezi řídícím hradlem a polovodičem technologie CMOS vznikne další vodivá vrstva, jenž je odizolovaná oxidem křemíku. Prahové napětí MOS tranzistoru určuje náboj na plovoucím hradle. K zápisu a mazání se využívá lavinového nebo tunelového průrazu. Při lavinovém jsou elektrony ve vodivém kanálu urychleny elektrickým polem na koncích (z angl. injekce tzv. horkých elektronů) a proletí tenkou vrstvou oxidu pod hradlem, metoda je velice rychlá, řádově v mikrosekundách.

Používá se tranzistor tzv. FAMOS (Floating-Gate Avalanche-Injection MOS). U tunelového průrazu se využívá manipulace pomocí velkých napětí a tenké vrstvy izolantu, přes níž pronikne elektron. Jedná se o kvantový jev tzv. Fawler-Nordheimův mechanismus. Metoda je pomalejší než předchozí, řádově v milisekundách. Oba mechanismy se využívají pro EPROM, EEPROM i FLASH. [13]

Mazatelná programovatelná paměť (EPROM – Erasable PROM)

Je programovatelná elektricky, před programováním je vymazána během 15 minut ultrafialovým zářením o vlnové délce 254 nm, které je generováno z rtuťové výbojky. Elektron se přes křemíkovou bariéru opět vrátí do polovodiče. Vymazaná paměť nabývá pro všechny bity hodnoty jedna. Při programování se pak využívá injekce horkých elektronů, ale i zvýšeného napájecího napětí cca 12 V. K naprogramování slouží přístroj, který má stejný název jako zkratka paměti. Trvanlivost dat záleží na teplotě. Mazání je však omezeno, řádově v desítkách opakování, takže je nevhodná pro časté mazání a relativně zastaralá. [13]

Elektricky mazatelná programovatelná paměť (EEPROM – Electrically Erasable PROM)

Zaznamenává se elektricky na libovolnou adresu bez předešlého vymazání, jde o podobný princip jako u RAM (hodnota 0/1), zápis je však o poznání pomalejší, než čtení. Paměť EEPROM funguje pomocí obou směrného tunelování elektronů. Buňka je složena z výběrového a paměťového tranzistoru, který obsahuje plovoucí a řídící hradlo. Nabízí neomezený zápis na jedné adrese. [13]

Paměť programovatelná a mazatelná (FLASH)

Stavbou je jednodušší, než EEPROM, skládá se z jedno tranzistorových buněk, umožňuje vysokou integraci. Nevýhoda spočívá v mazání paměti sice elektricky, ale jako celku po sektorech, či blocích, které je pomalé kolem jedné sekundy. Na druhou stranu zápis probíhá rychle elektricky na libovolné adrese. Kombinuje obě výše zmíněné technologie, lavinovou injekcí a záporným nabitím hradla dochází k naprogramování, tunelováním elektronů u ní dochází k vybití hradla a vymazání buňky. Paměť se dělí na sektory, který lze promazat a naprogramovat. [13]

Paměti EEPROM a FLASH mohou být zapájeny do zařízení, jelikož se dají programovat i elektricky až ve výsledném zařízení. Obě mají vedle čipu integrovaný sekvenční obvod, neboli řadič, který řídí práci s pamětí. Zvýšené napětí je také generováno přímo z nábojové pumpy, takže zvenku postačí běžné napájecí napětí.

Čtení z paměti je podobné jako u RAM, zápis a mazání se liší podle typu a výrobce. Zápis je pomalejší, avšak obsahuje posuvný registr, který postupně zavádí data do paměti. Klasické EPROM, EEPROM, a FLASH bývají až desetkrát pomalejší, než SRAM. Avšak pro synchronní EEPROM a FLASH to neplatí, jelikož paměti jsou vybaveny čítačem sloupcových adres a vydávání dat je synchronizováno s rytmem hodinových pulsů. Starší matice buněk je uspořádána jako NOR FLASH, jedná se o paralelní zapojení tranzistorů na sloupcový vodič, každá buňka je zapojena na sloupcový vodič a zem. Novější sériové spojení tranzistorů NAND FLASH, je jednodušší uspořádání s vyšší hustotou integrace, ale je pomalejší než NOR FLASH. [13]

1.2.3 Hardwarově akcelerované karty

Program bude realizovaný na jednom ze dvou typů desek se síťovými kartami a hradlovým polem.1.9 První je deska se síťovou kartou NFB (dříve COMBO) – 200G2QL 1.10 od Netcope technologies [1] s programovatelným hradlovým polem

[1] [9]

Virtex UltraScale+ od firmy Xilinx [2], které může být programované buď P4, nebo HDL programovacím jazykem. Propustnost komunikační sítě (mezi hostitelským počítačem a technickým vybavením desky) je až 200 Gb/s je zajištěna pomocí dvou optických vláken s rozhraním typu QSFP28 a 3x16 PCI sběrnicí, která má šestnáct paralelně zapojených konektorů. Síťová karta pro komunikaci s počítačem používá rozhraní Ethernet 10G, 40G, 100G. Programovatelné hradlové pole má k dispozici 1 724 000 logických buněk [16]. Předchozí verze čipu Virtex-7 od již zmiňované firmy disponuje množstvím 580 480 logických buněk. Deska se síťovou kartou a hradlovým polem má podobné vlastnosti jako novější verze.

	Netcope NFB-200G2QL Xilinx Virtex Ultrascale+ vu7p	Netcope NFB-100G2Q Xilinx Virtex 7 xc7vh580t
Systémové logické buňky	1 724 000	580 480
Flip-Flop	1 576 000	725 600
LUT tabulky	788 000	
celková velikost blokové RAM (b)	50 600 000	33 840 000
URAM	90 Mb	
SRAM	3x75 Mb / 3x288 Mb	3x75 Mb každá
PCIe Gen3x16	2	2
100G Ethernet	2	1
maximální počet samostatně zakončených vstupů/výstupů	832	600
Propustnost k RAM	200 Gb/s	100 Gb/s

Obr. 1.9: Porovnání karet zmíněných v textu [9] [18]

1.3 Jazyk VHDL

Pro naprogramování zvolených funkcí na hradlovém poli, použijeme jeden ze dvou programovacích jazyků pro tento účel, které se jmenují VHDL a Verilog, SystemVerilog. Jako všechny ostatní jazyky je rozlišujeme podle výskytu největšího používání. Přičemž Verilog je spíše využíván v Asii a USA, oproti VHDL, který je používán hlavně v Evropě [7].

[2] [18]

Obr. 1.10: FPGA NFB-200G2QL Virtex UltraScale+ [8]

Název jazyku VHDL vznikl jako akronym VHSIC z Very High-Speed Integrated Circuit v akronymu VHDL z VHSIC Hardware Description Language. Pod zkratkou z počátečních písmen anglických slov se skrývá jazyk, který zprostředkovává komunikaci s číslicovými systémy složenými z velmi rychlých integrovaných obvodů. S jeho pomocí konfigurujeme přímo hradlová pole (hradla, klopné obvody). Počítáme tedy spíše s logickou strukturou programu oproti jazykům C a C++, vzhledem k součástkám, s kterými jazyk spolupracuje. VHDL je jazyk vysoké úrovně, který je specifický svou přenositelností, dobrou srozumitelností, jenž využívá překladač kódu. [13]

Jazyk vznikl pro účely armády pod projektem se stejným názvem jako akronym VHSIC v roce 1981, na vývoji poté spolupracovaly i firmy IBM, Intermetrics a Texas instruments, inspirovali se již vzniklým jazykem ADA. V roce 1987 byl organizací IEEE (angl. Institute of Electrical and Electronics Engineers) poprvé vydán standard jazyka. Měl by být po pěti letech pravidelně revidován, což se také snaží dodržovat. [13]

Stěžejními architektonickými prvky jazyku VHDL jsou entita a architektura. V entitě definujeme velikost a typ signálů, které použijeme, jestli se jedná o vstupní, výstupní, či kombinovaný. Přičemž doporučená maximální velikost nabývá hodnoty 512 bitů. Avšak lepší je pracovat s menšími prvky např. o velikosti 64 – 128 bitů, aby mohl být algoritmus rychlejší. Dalším stavebním kamenem je architektura, která definuje logické operace s předem definovanou entitou, popřípadě s dalšími deklarovanými proměnnými. Program by měl být rozdělen do co nejjednodušších prvků,

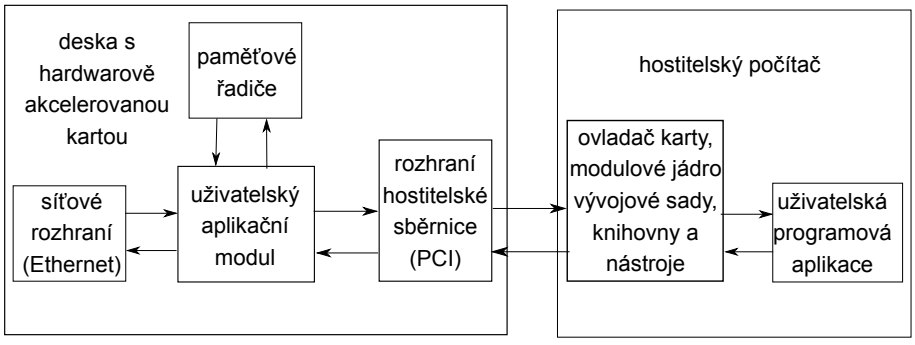
aby se eliminovalo zpoždění. Zápis probíhá buď synchronně, zároveň s hodinovým signálem, nebo asynchronně, nezávisle na hodinách. [17]

1.4 Vývojový nástroj od firmy Netcope (NDK – Netcope Development Kit)

Vývojový nástroj obsahuje technicky zrychlenou kartu s čipem FPGA a programové vybavení pro konfiguraci přes hostitelský počítač, které běží na CPU. Komunikace mezi dvěma výše zmíněnými částmi probíhá pomocí vysokorychlostní PCI sběrnice. Programová část NDK obsahuje řadiče, knihovny a nástroje, DMA (direct memory acces) zprostředkovává komunikaci a má přístup k registrům, umožňuje rozesílání dat a kontrolu běžící aplikace v hradlovém poli z hostitelského počítače. Běžné aplikace, které jsou navržené pro zpracování procesorem mají pouze jednu část, zatímco zrychlená aplikace pro programovatelné hradlové pole se skládá ze dvou částí, z programové části, která běží v připojeném počítači a ze zrychleného jádra, jenž se nachází na čipu FPGA. [10]

Hlavní součásti firemního výrobku: [10]

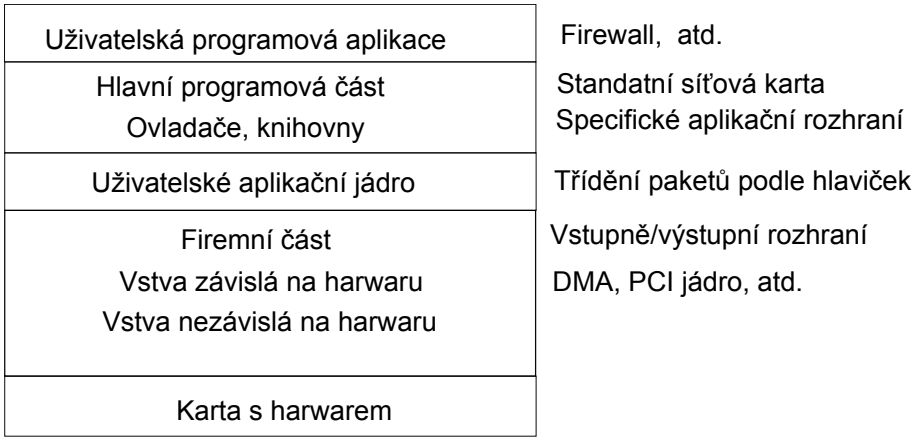
- **vstupní a výstupní rozhraní** – pro přijímání a odesílání paketů, liší se v šířce pásma a technologiích
- **přístup ze softwarové aplikace** – obsahuje programové řešení komunikující skrz PCI sběrnici a zprostředkovává čtení a zápis ze zrychleného jádra, které je umístěné v obvodu s hradlovým polem.
- **rychlý přenos dat** – velice rychlý přenos paketů je zajištěn pomocí DMA, které je obsaženo v rozhraní síťové karty. Používá se i standardní síťové nebo specifické aplikační rozhraní.



Obr. 1.11: Diagram zobrazující spolupráci technického a programového vybavení [10]

Zdrojový archiv obsahuje podadresáře aplikační, dokumentační a ndk. Aplikační obsahuje konfigurační soubory, uživatelské aplikační jádro a simulační testovací ša-

blonu, v dokumentační je uložena dokumentace a v ndk jsou zdrojové soubory, IP jádro a seznam síťových souborů. Poslední zmiňovaný adresář zahrnuje i skripty, které jsou používány k syntéze a simulaci NDK. [10]



Obr. 1.12: NDK vrstvy [10]

1.4.1 Aplikační část

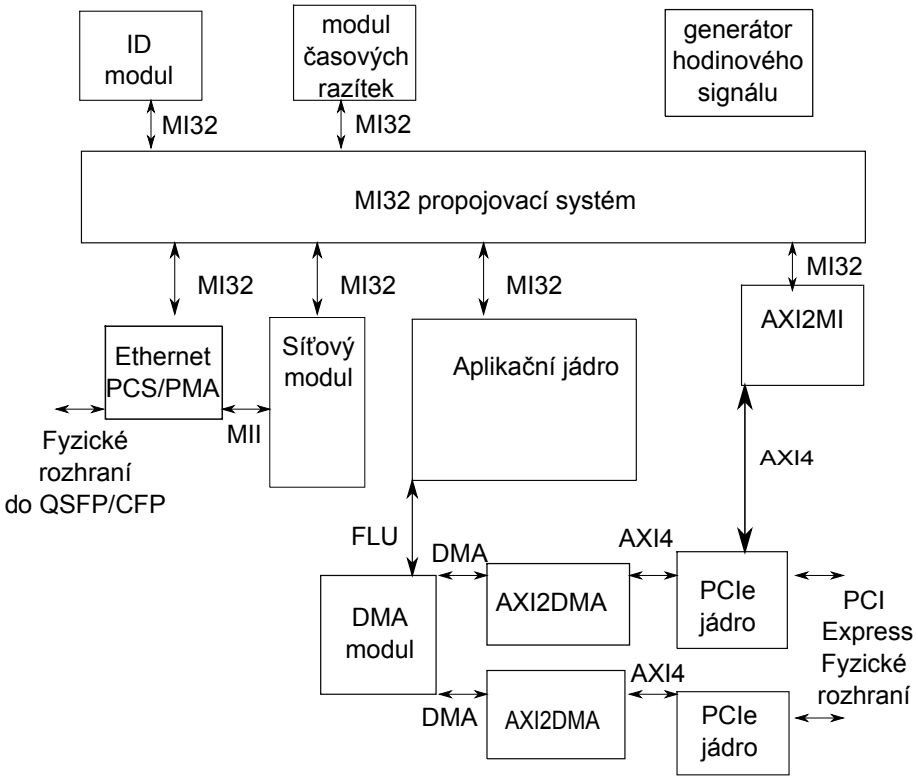
Obsahuje podsložku s označením nic, přičemž X značí rozhraní, jenž jsou dostupná pro konfiguraci sítě. Například název nic_10g8 značí osm konektorů s propustností každého 10 Gb/s. Každá tato podsložka má výše uvedený obsah, konkrétně: Konfigurační soubor (config), který obsahuje hodinový signál, nebo DMA konfiguraci kanálů. Vymezující soubor (constr), který bývá ve výchozím stavu prázdný. Simulační soubor, jenž zahrnuje testovací šablonu pro uživatelské aplikační jádro (sim). Zdrojový soubor (src), v němž je vložená architektura uživatelské aplikace. Vytvoření souboru (Makefile), což je automatický nástroj pro simulaci a syntézu, spustitelný skrz Linuxový program Vytvoř nástroj (Make utility). (Modules.tcl) Konfigurační soubor s přidanými zdrojovými kódy pro syntézu a simulaci. Strom zařízení (DevTree.tcl) řeší konfigurační adresový prostor pro uživatelské aplikační jádro. Vivado (vivado.tcl) s předdefinovanými cestami a parametry pro Vytvoř nástroj. Vytvoř (build) je adresář, který je určený pro výstupy simulace a syntézy, obsahuje výstupy simulace, syntézy (i generované z uživatelského rozhraní programu Vivado) a FPGA programové části. [10]

Adresář ndk obsahuje dvě podsložky, první je složka Karta (<card>) s konkrétními soubory pro dané hradlové pole a druhá Obecná (common) jež obsahuje běžné zdrojové soubory. Výše zmíněná složka karta dále obsahuje archivy s Konfiguračními soubory (config), není doporučeno je měnit. Vymezující soubor (constr), v němž se definuje lokalizace pinů a specifické přiřazení pro programovatelné hradlové pole.

Jádra (cores), v němž se nachází IP jádro. Složky s individuálními komponentami (dcp), se zdrojovými kódy (src), konfigurační soubory, tcl skripty (tcl). Poslední jmenované jsou používány během syntézy a je možné je změnit. Vytvoření souboru (Makefile.ndk) jedná se o konfigurační soubor pro automatickou syntézu.

1.4.2 Prostředí firemního výrobku

Prostředí je primárně určeno pro rychlý vývoj síťových aplikací na programovatelném hradlovém poli. Skládá se z jednotlivých částí jako Síťová "DMA". Moduly jsou napsány v HDL jazycích (VHDL nebo Verilogu) či jsou vytvořené hlavním generujícím IP protokolem od firmy Xilinx (Xilinx IP Core Generator). Předpokládá se, že se vloží program do Aplikačního jádra a nebude nutné pozměňovat samotnou architekturu NDK. Aplikační jádro může například pozměňovat každou procházející datovou jednotku mezi síťovým a DMA modulem. ID modul obsahuje základní informace o firemním výrobku a jeho parametrech.



Obr. 1.13: NDK blokový diagram firemního výrobku [10]

1.4.3 Popis jednotlivých modulů

Aplikační jádro

Aplikační jádro je určeno pro vložení uživatelské aplikace. Je připojené k Síťovému a DMA modulu (pro přenos dat mezi sítí a hostitelským počítačem), zároveň k MI32 propojovacímu systému, aby Aplikační jádro mohlo být ovládáno z hostitelského počítače.

Při příjmu síťového provozu jsou ethernetové rámce přeneseny do Aplikačního jádra skrz FLU sběrnici. NDP hlavičky obsahují více informací o paketu, jsou přenášeny skrz RámecSpojovou sběrnici (FrameLink bus) a jsou použity pro přenosy (DMA) rámců do programu.

Uvnitř aplikačního jádra se nachází NDP modul, jenž vkládá hlavičky, které jsou dány před originální balíčky dat. NDP hlavička může přijít dříve, než samotná datová jednotka, nikdy ne naopak, takže se nemusí dostavit ve stejném hodinovém cyklu. Záleží na implementaci NDP generátoru hlaviček (PACODAG), platnost hlaviček je určena signálem FrameLink/FLU rozhraní zdroj připraven (SRC). NDP hlavičky jsou přenášeny FLU Plus sběrnicí do DMA modulu, z něhož jsou následně odeslány pomocí DMA přenosů do RAM hostitelského počítače. Uživatel určí číslo DMA kanálu, který bude použit pro přenos pomocí nastavení kanálu (CHANNEL) signálu ve FLU Plus sběrnici.

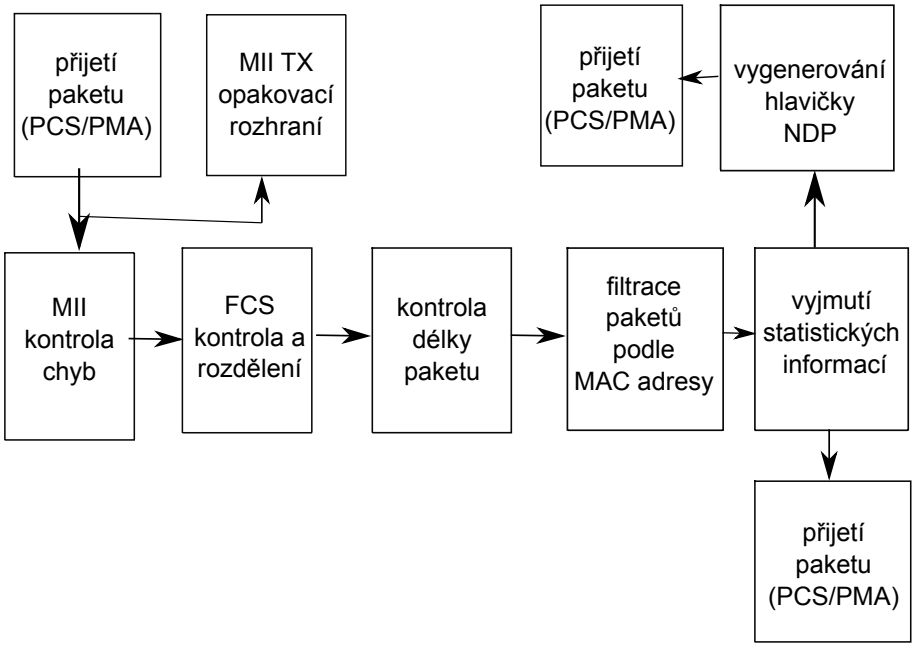
Uživatelská aplikace je schopná přistupovat k balíčkům dat skrz FLU a FLU Plus sběrnici a k NDP hlavičkám na FrameLink sběrnici. Datové jednotky můžeme měnit (včetně NDP hlaviček), zahazovat je, či přidávat další. Změníme-li délku datového úseku je nutné změnit i v NDP hlavičce Velikost Segmentu (Segment Size) a stejně se postupuje při změně NDP hlavičky, je nutné přepsat i Velikost Hardwaru/Segmentu (Hardware Size).

Odesílají-li se balíčky dat směrem do sítě, Ethernetové rámce jsou přenášeny z hostitelského počítače pomocí FLU Plus sběrnice z DMA modulu pomocí definovaného signálu kanálu (CHANNEL) do programovatelného hradlového pole. Poté jsou data přeneseny pomocí FLU Plus do Síťového modulu. V tomto případě (CHANNEL) signál Kanálu značí rozhraní, kterým má být datová jednotka přenesena. NDP hlavička pro přenos musí být dlouhá 8 bajtů. MI32 sběrnice je připojená k Aplikačnímu jádru a umožňuje snadné ovládání z hostitelského počítače, jelikož přenáší hodnoty adresového registru mezi firemním zařízením a programem v hostitelském počítači.

Síťové moduly

Síťový modul zajišťuje přenos Ethernetových rámců mezi PCS/PMA modulem skrz MII rozhraní a Aplikačním jádrem pomocí FrameLink, FLU a FLU Plus sběrnice. MI32 sběrnice je připojena na stejnojmenný propojovací systém, který zajišťuje kontrolu toku datových jednotek a sleduje statistiky odeslaných a přijatých balíčků dat. MII rozhraní mezi síťovým rozhraním a PCS/PMA modulem je definováno pomocí IEEE 802.3 standardu a může mít formu XGMII (10 gigabitového nezávislého rozhraní média), XLGMII (40 gigabitového nezávislého rozhraní média) či CGMII (100 gigabitového nezávislého rozhraní média).

Když jsou přijímány bloky dat ze sítě, jsou datové jednotky přemístěny pomocí PCS/PMA modulu skrz MII rozhraní do Síťového modulu. PCS/PMA modul přenáší pakety mezi síťovými rozhraními skrz GTx transceiver a Síťovým modulem pomocí MII sběrnice. V síťovém modulu jsou dále datové jednotky filtrovány dle MAC adres. Získávají se i další informace z metadat z nichž jsou generovány NDP hlavičky a jsou odesílány do aplikačního jádra FrameLink sběrnicí. Následně se paket odešle skrz FLU sběrnici do Aplikačního jádra. Současně je původní datová jednotka odeslána na MII TX Opakovací rozhraní (Repeater interface), pokud síťový modul pracuje v Opakovacím režimu (Repeater mode), z rozhraní pokračuje balíček dat zpět do sítě.



Obr. 1.14: Příjem paketů ze sítě do Síťového modulu [10]

Síťový modul provádí i kontrolu kontrolního součtu bitů (FCS, CRC), po kontrole jsou bity odstraněny. Následně MII zkontroluje chyby v délce datové jednotky. Ve

výchozím nastavení jsou všechny bloky dat, které neprojdou kontrolou bezchybně, zahozeny, nastavení se dá změnit v „ndktool cnt tool“. Pomocí předchozího zmíněného nástroje lze sledovat statistiky přijatých, vadných, či filtrovaných datových jednotek.

Modul podporuje 4 módy filtrování paketů podle MAC adres. Módy jdou nastavit pomocí uvndktool rx tool. Mód 0 je nazývaný promiskuitní, při něm jsou všechny pakety přeneseny do Aplikačního jádra, není kontrolována MAC adresa. V módu 1 je nastavena tabulka povolených MAC adres, podle které se rozhodne, které poputují do Aplikačního jádra. Ostatní jsou zahozeny. Další mód je podobný jako předchozí, jediný rozdíl spočívá ve faktu, že i všesměrové pakety pokračují do Aplikačního jádra. Poslední mód 3 je obohacen o skupinové pakety, které se opět odešlou do Aplikačního jádra.

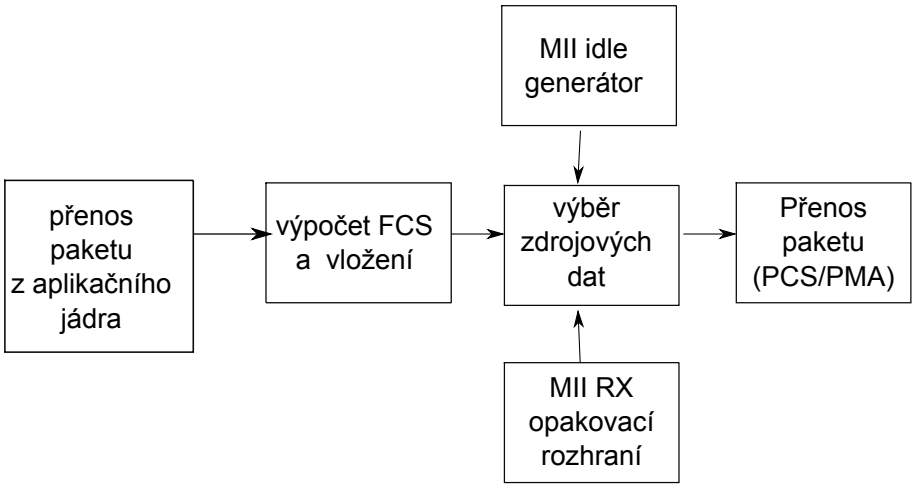
NDP Generátor hlaviček a PACODAG Modul

Délka paketu je z něj vyčtena a je odeslána do PACODAG modulu (PAcket COntrol DAta Generator), který generuje NDP hlavičky, jenž následně putují zpět do Síťového modulu. Z něhož jsou odeslány FrameLink sběrnicí do Aplikačního jádra.

Přenos dat do sítě je závislý na nastavení ndktool tx tool, ve výchozím nastavení modul pracuje v režimu „OBUF“, kdy datové jednotky, které jsou přijaty z aplikačního jádra jsou přeneseny do sítě skrz FLU Plus sběrnici a data z MII opakovacího rozhraní jsou zahozeny. V síťovém modulu je spočítán (FCS) kontrolní součet a pole (FCS) je přidáno k paketu. Dále datová jednotka pokračuje do PCS/PMA modulu a následně je odeslán do sítě. Je možné nastavit další dva způsoby přenosu datových jednotek. Prvním je tzv. „Idle“, který odesílá pouze „Idle“ data z MII a datové jednotky z aplikačního jádra, ale z MII opakovacího rozhraní přeposílány nejsou. Druhým je Opakovací mód, jak už napovídá jeho název, data jenž byla přijata ze sítě, jsou přeposílána skrz MII TX Opakovací rozhraní. Statistiky odeslaných paketů se dají zobrazit v ndktool cnt tool, když síťový modul pracuje v „OBUF“ či Opakovacím módu.

DMA modul

Poskytuje oboustranný přenos datových jednotek mezi programovou aplikací, která běží na hostitelském počítači a NDK firemním nástrojem, aniž by byla použita Centrální procesorová jednotka hostitelského počítače. DMA modul přenáší informace velmi efektivně pomocí paralelní implementace mnoho vláknové PCI sběrnice třetí generace, která je synchronizována a zajišťuje maximální propustnost. DMA modul zahajuje a provádí obou stranný přenos mezi vyrovnávacími paměťmi. Je založen na



Obr. 1.15: Operace, které probíhají v Síťovém modulu při odesílání paketů do sítě [10]

principu dvou kruhových vyrovnávacích pamětí pro oba směry DMA kanálu, jedna se nachází v operační paměti počítače a druhá v FPGA.

Modul časových razítek

Modul časových razítek pro NDK se skládá z 64 bitového registru a je obnovován každé 3-10 ns, což je dáno frekvencí hodinového signálu pro daný modul. Vyšších 32 bitů značí sekundy a nižší nanosekundy. Časový interval odpovídá koordinovanému časovému pásmu UTC. Pro vytvoření časových pulzů se používá krystalový oscilátor umístěný na kartě, nebo v externím zdroji. Pro synchronizaci měření času je doporučené užití jednotky počet pulsů za sekundu PPS (Pulse Per Second). Vhodným zdrojem synchronizace PPS signálu je GPS příjmač. Časové známky se nastavují v ndktool time tool.

MI32 propojovací systém

MI32 řídí čtení a zápis požadavků od hostitelského počítače, které mají být provedeny v modulech firemního výrobku. Výběr modulu závisí na adrese MI32 v požadavku. Modul shromažďuje odpovědi na požadavky, které jsou pak poslány zpět do počítače. Systém priorit řešení je založen na multiplexování každé žádosti, která je vyhodnocována podle adresy. Propojovací systém rozděluje adresní prostor napříč NDK firemními moduly a řídí požadavky.

Modul pro vytváření hodinového signálu

Modul pro vytváření hodinového signálu slouží všem ostatním modulům. Veškerý hodinový signál je generovaný oscilátorem umístěným na kartě a dá se k němu přistupovat přes signál CLKIN (125 MHz), který je spojený se vstupem MMCME2. Hodinový signál může být násobený, nebo dělený ze základních 1400 MHz a je rozváděný do obvodů FPGA pomocí hodinových vyrovnávacích pamětí BUFGs.

PCI-E vedoucí ke koncovému bodu

PCI-E jádrový modul je tvořen komunikačním rozhraními pro PCI sběrnici třetí generace a AXI4-Streamem, mimo sběrnici je tvořen Xilinx IP Core Generátorem. Rozhraní PCI sběrnice slouží k fyzické komunikaci s porty dané sběrnice. AXI4-Stream zajišťuje přenášení kódu. Konvertory zdrojového kódu do AXI4 pro moduly DMA a MI32 pro jeden směr komunikace jsou připojeny na jednu sběrnici a na druhou sběrnici konvertory pro druhý směr. XI2MI ze zdrojového kódu je konvertováno na AXI4-Stream ve sběrnici příslušného modulu a naopak, která je dále připojena k PCI–E modulu přes sběrnici AXI4–Stream.

1.4.4 Dva typy komunikačních rozhraní firemního nástroje

Komunikačním rozhraním pro přenos malých objemů dat a nepravidelných toků je protokol MI32. Zajišťuje konfiguraci modulů v programovatelném hradlovém poli a čtení informací z modulů v hostitelském počítači. Komunikace probíhá pomocí operací čtení a zápis, které jsou kontrolované z počítače. Moduly a registrové sety jsou přístupné přes adresy. Druhým naopak datovým rozhraním pro přenos velkých objemů dat s nepřetržitým provozem jsou FrameLink, FLU a FLUPlus sběrnice. První zmíněná obstarává přenos NDP hlaviček a rámců. Druhá zajišťuje přenos paketů do programovatelného hradlového pole a třetí je pro opačný směr.

FrameLink sběrnice

FrameLink sběrnice je dvoubodové vysokorychlostní rozhraní, slouží k přenášení NDP hlaviček. Data můžou být rozdělena na části. Délka dat přenášených ve sběrnici je 16, 32, 64 a 128 bitů. Dále obsahuje signály, které indikují připravenost odesílajícího a příjmajícího, začátek a konec paketu, pozici posledního bajtu v části. Přenos dat je orientován v sestupném pořadí bitů, od největšího po nejnižší př. (15 downto 0). Většina signálů v této sběrnici je aktivních se sestupnou hranou hodinového signálu.

FrameLinkUnaligned/FLU sběrnice

Může zpracovávat data o délce typicky 256, nebo 512 bitů. Signály se mění s náběžnou hranou hodinového signálu. Nalezneme v ní signály, které opět značí připravenost odesílajícího a příjmajícího, dané signály mohou nabývat hodnoty 0 a 1. Přičemž 1 značí, že jsou obě strany připraveny a může dojít k přenosu paketů ze síťového do aplikačního modulu. Dále obsahuje signály, který značí začátek (SOP–start of packet) a konec paketu (EOP–end of packet), které se mění z 0 na 1, když nabývají hodnoty 1 indikují, že v příslušném balíku dat se nachází začátek, nebo konec paketu. Přesnou pozici začátku nebo konce značí signály (SOP_POS–packet start position) a (EOP_POS–packet end position). Přičemž začátek je značen po 64 bitech v signálu o velikosti 3 pro signál dat s velikostí 512 bitů, takže například binární hodnota 001 značí, že se začátek nachází na pozici 64. Přesný konec paketu je zaznamenán v 6 místném poli opět pro datové pole 512 bit a je udáván po bajtech. Například binární hodnota 010011 značí, že konec paketu se nachází na 19 bajtu od začátku právě přenášeného balíčku dat. Pro FLU Plus sběrnici to funguje obdobně, akorát pro opačný směr.

1.5 Vývojové prostředí

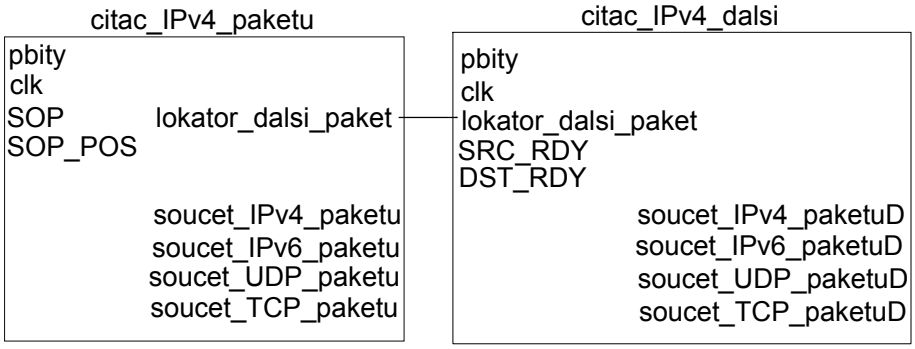
Celý program jde odsimulovat pomocí vývojového prostředí Vivado pro hradlová pole od firmy Xilinx, do něhož se zapíše kód pomocí zvoleného VHDL jazyka. Kdy si pro simulaci navrhneme „testovací sadu“ (testbench), kde si nadefinujeme signály, pomocí kterých budeme náš kód podrobovat zkoušce. Při simulaci se zobrazí testovací signály a jejich změny v čase. Záleží na jaký časový úsek si změny vstupních signálů navrhneme a poté si můžeme prohlédnout, zda výstupní signály odpovídají očekávání. Pomocí syntézy se odhadne kolik prostředků bude potom z FPGA použito, tímto způsobem se dá optimalizovat návrh. Posledním krokem po úspěšném odsimulování a vhodném využití prostředků FPGA se přejde k implementaci, která již nahraje navržený algoritmus do zařízení, kde se rozhoduje o umístění komponent na hradlové pole.

2 Výsledky studentské práce

2.1 Čítač paketů

Analýzou sítě je čítání prošlých IPv4, IPv6, UDP a TCP paketů, které probíhá synchronně. Což znamená, že čítání je citlivé na náběžnou hranu hodinového signálu. Řešení v jazyce VHDL je realizováno pomocí signálů již definovaných ve firemním výrobku a použití daných sběrnic. Konkrétně ve FLU sběrnici, která umožňuje pracovat s daty ze sítě pomocí logického vektoru(std_logic_vector) o velikosti 512, nebo 256. Využijeme signálů *SOP* a *SOP_POS* s jejichž pomocí je nalezen počátek paketu, dále je realizován přesun na pole s informací, zda jde o verzi s IPv4, nebo s IPv6 paketem. Následně se zjišťuje zda IP paket obsahuje vybraný UDP, či TCP datagram. Pokud najde jeden z předchozích zmíněných protokolů příslušný čítač zvýší hodnotu o jedna.

Program se skládá ze dvou komponent.2.1 Jedna pro momentálně příchozí paket se signálem, který značí začátek paketu. Druhá pro paket, který přijde s následujícím hodinovým signálem, který již neobsahuje signál začátek paketu. Příchozí data obsahují začátek paketu, pokud signál *SOP* nabývá hodnoty jedna. *SOP_POS* indikuje přesnou pozici počátku příchozího paketu, informace je uložena opět v std_logic_vector, což je v postatě pole hodnot, pozice je uložena v násobcích 8 bajtů. Což znamená, že pokud hodnotu vynásobíme 64, dostaneme začátek paketu v bitech.



Obr. 2.1: Schéma návrhu

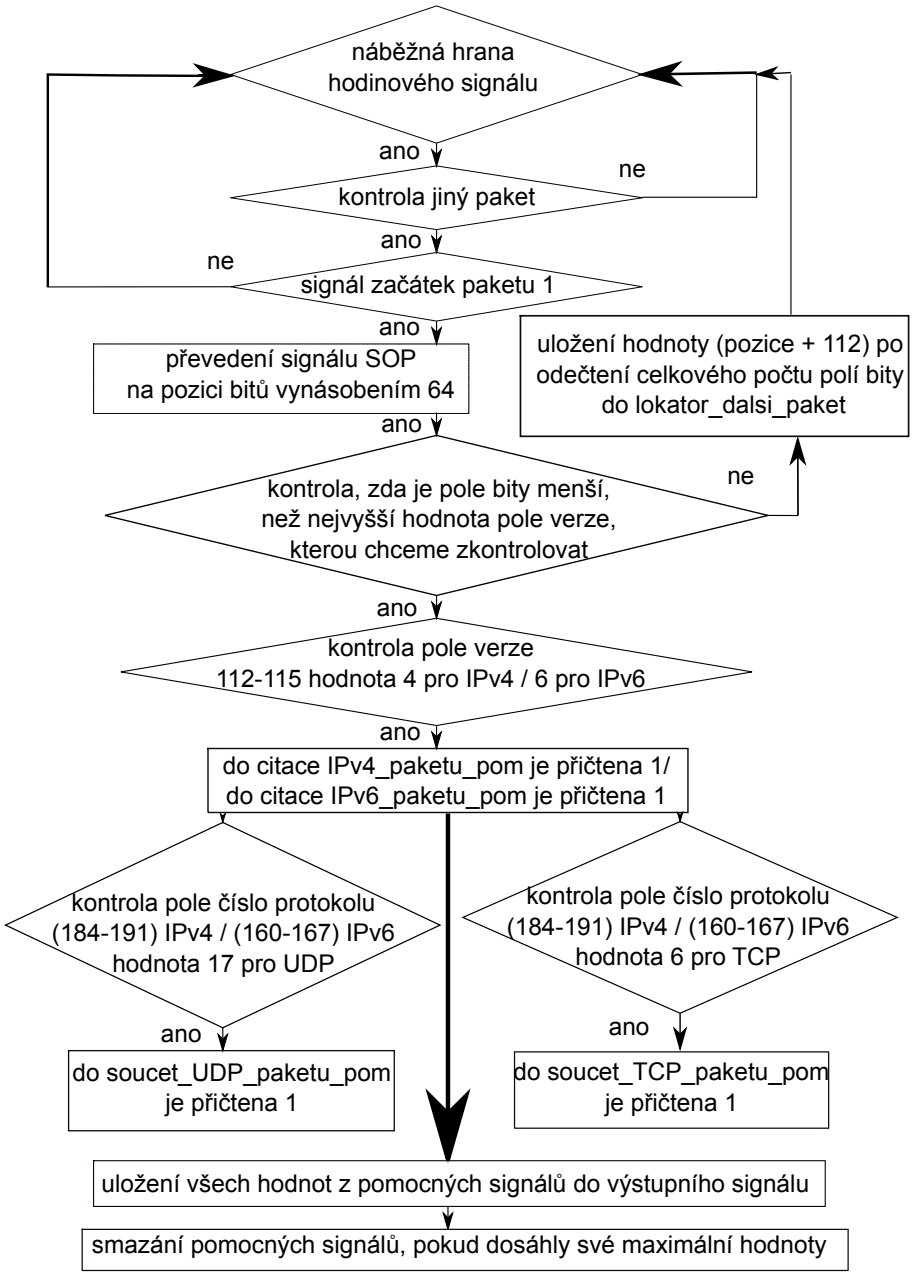
Začátek paketu se může nacházet v násobcích 64 bitů v podstatě kdekoli, nemusí se tedy námi požadované pole s hodnotou verze protokolu, či číslo protokolu nacházet již v paketu, který obsahuje signál *SOP*. Proto opět kontrolujeme to samé v další komponentě po přepočítání pozice, kde by se mělo nacházet námi požadované pole. Daný signál si komponenty předají. Druhá komponenta provede kontrolu, zda je vysílač i příjmce připraven, což vidíme v signálech *SRC_RDY* a *DST_RDY*, které

musí nabývat hodnoty 1, potom dochází k přenosu paketů. Dále se kontroluje, jestli se příchozí paket neshoduje s právě přičteným. V případě, že se program dostane až ke kontrole pole bity na příslušné pozici a v některém kroku hodnoty nejsou shodné, pak se nepřičte v čítači hodnota pro čítání paketů s IPv4, IPv6, UDP či TCP.

Komponenta obsahuje vždy vstupní a výstupní signály. Do obou komponent přichází hodinový signál pojmenovaný clk, dále pak datový signál bity, který obsahuje informace, které přišly ze síťového modulu. Výstupní signály pro obě komponenty se shodují ve čtyřech signálech pro čítače paketů pojmenovaných podobně *soucet_IPv4_* paketu pro první *citac_IPv4_paketu*, pro druhou komponentu *citac_IPv4_* dalsi je přidáno nakonec pouze písmeno D *soucet_IPv4_paketuD*. Dále první komponenta obsahuje vstupní signály zmíněné již výše, druhá nápodobně. Signál, který je mezi nimi předáván se nazývá *lokator_dalsi_paket*.

Bude uvedena konkrétní funkce komponenty *citac_IPv4_paketu*. 2.2Jak již bylo zmíněno, kontrola probíhá s nástupnou hranou hodinového signálu. Dále je zkontrolováno, jestli nečítá již jednou načtený paket, který jen je déle přítomen např. přes dva hodinové signály. Následně kontroluje, zda signál, který indikuje začátek paketu nabývá hodnoty jedna. Potom probíhá kontrola, zda po přepočítání pozice pro přesunutí na odečtení hodnoty, nepřesahuje rozsah pole s daty. Pokud přesahuje, pole pozice se přepočítá pro začátek pole verze v dalším paketu a uloží se do výstupního signálu *lokator_dalsi_paket*. Pokud k přetečení pole nedojde program přejde ke kontrole v polích 112 až 115 pro IPv4 hledá hodnotu 4 a pro IPv6 číslo 6. Pokud nalezne, do čítače inkrementuje o jedničku. Dále se konroluje v polích 184–191 pro IPv4, zda obsahuje protokol UDP, nebo TCP a pro protokol IPv6 v polích 160–176. Opět pokud nalezne shodu čítače jsou inkrementovány o jedničky. Dále jsou hodnoty přiřazeny do výstupních portů a ještě proběhne kontrola přetečení logických vektorů pro součty.

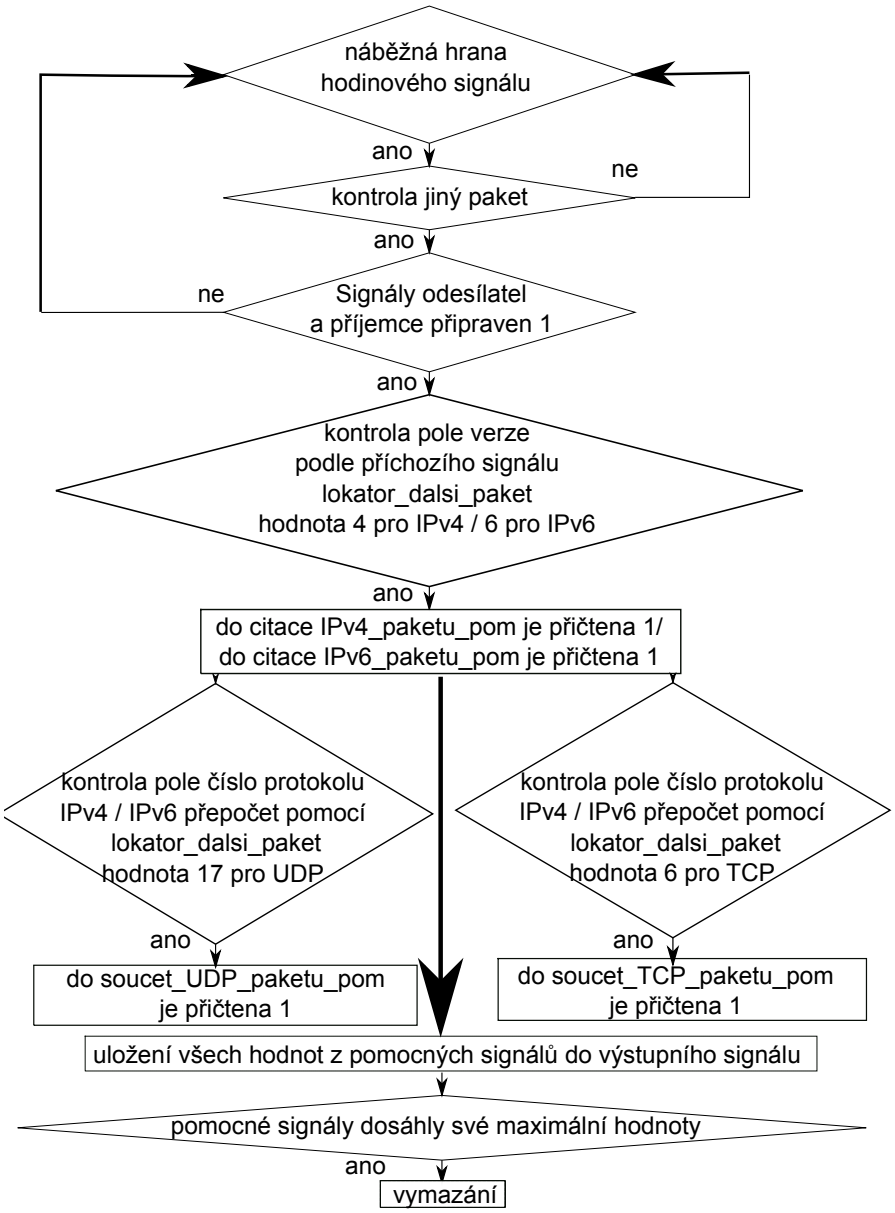
Funkce druhé komponenty *citac_IPv4_dalsi* je obdobná.2.3 Program se spustí s náběžnou hranou hodinového signálu, dále se opět zkontroluje, zda jenom nevzal předchozí již čítaný paket. Poté se zkontroluje, zda signály příjemce a odesílatel připraven na hodnotě jedna, pomocí příchozího signálu *lokator_dalsi_paket* nastane ke kontrole IPv4 a IPv6 protokolu, pokud nalezne shodu, přičte jedna posune se na kontrolu pole číslo protokolu, které je opět odlišně umístěné pro IPv4 a IPv6. Pokud najde shodu opět přičte 1. Nakonec jsou pomocné signály přiřazeny do výstupních portů. Probíhá kontrola, zda již nenabývá pole s pomocnými součty maximální hodnoty, v případě, že ano. Signály jsou promazány.



Obr. 2.2: Vývojový diagram komponenty citacIPv4paketu

2.1.1 Simulace

Pro simulaci je potřeba si zadefinovat tzv. Testbench, testovací šablona, do které vložíme entity, které chceme v simulaci testovat a propojit. Přejmenujeme je na komponenty. Dále si deklarujeme signály, které půjdou do vstupních, výstupních portů, popřípadě mezi komponantami. Signály následně přiřadíme portům pomocí klíčového slova PORT MAP. Potom si nadefinujeme přímo vstupní signály, které se budou měnit s nějakou časovou periodou. Nemusíme definovat časový úsek, ale

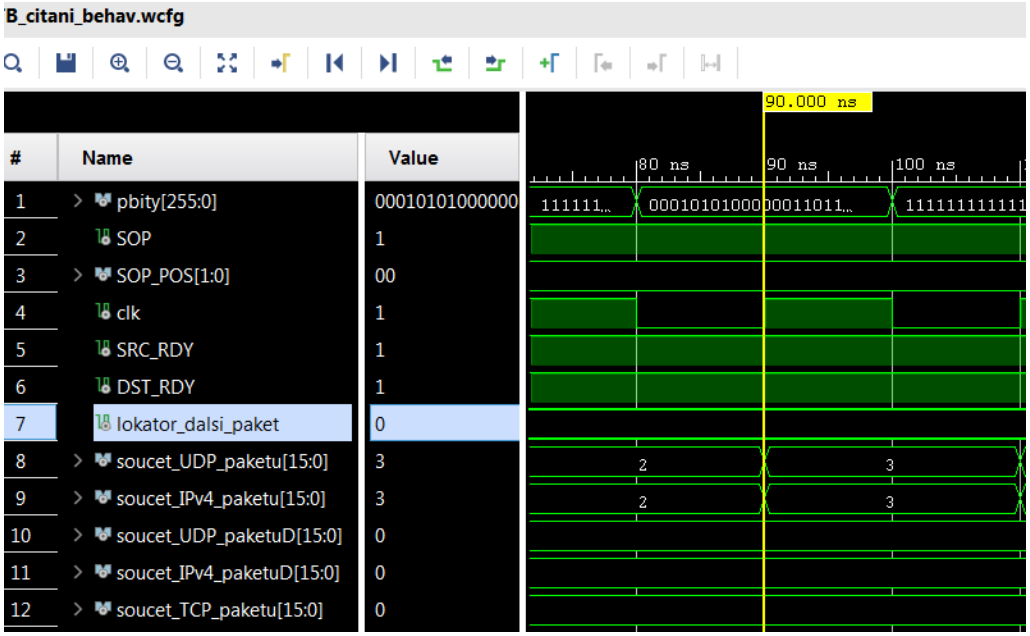


Obr. 2.3: Vývojový diagram komponenty citacIPv4dalsi

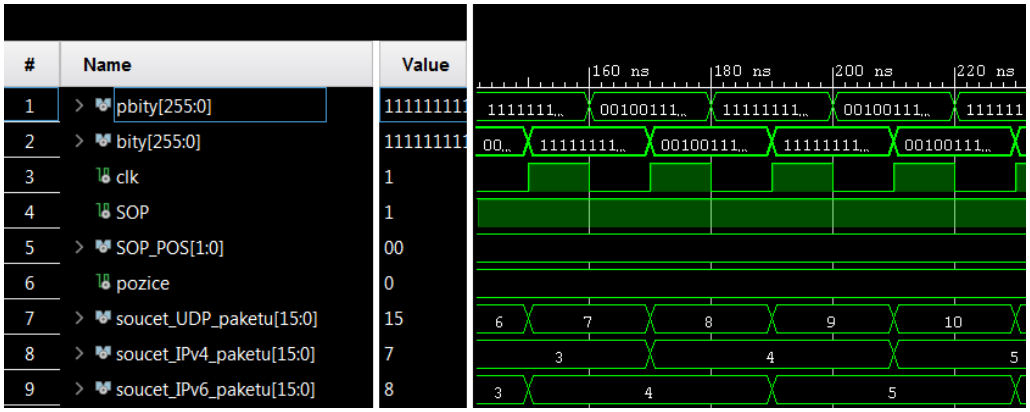
vzhledem k synchronnímu návrhu kódu je to vhodné. Simulace je spíše teoretická zkouška kódu. Až syntézou se dá zjistit, jestli je možné kód aplikovat na programovatelné hradlové pole.

Pro simulaci jsou používány pakety, které byly odchyceny v programu Wireshark, zkopírovány, obráceny (vzhledem k sestupné doporučené definici signálů) a zkráceny do logického vektoru pbity. Ze simulací je patrné 2.4 2.5, že komponenta *citac_IPv4_paketu* funguje správně a signály jsou přiřazovány včas výstupním por-

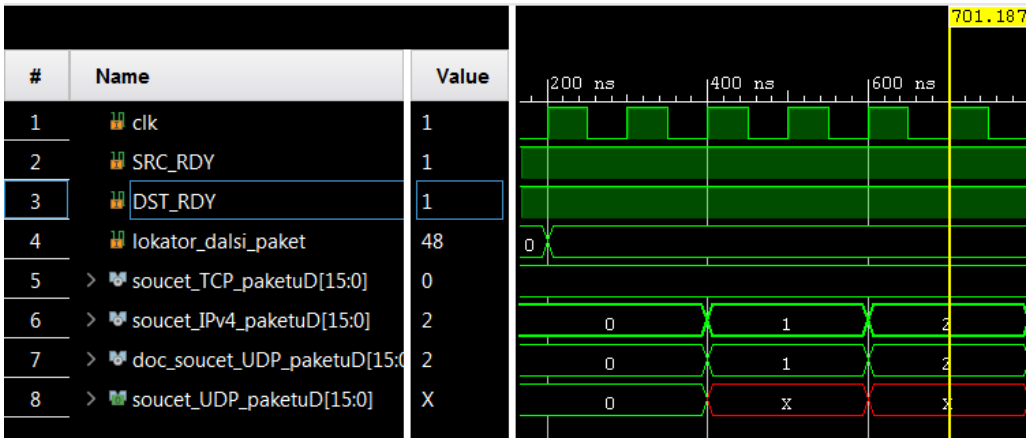
tům. U druhé komponenty program nestíhá přiřazovat hodnotu do výstupního portu *citac_UDP_paketuD*, avšak v pomocném signálu je hodnota včas. 2.6 Nepomohlo ani přemístění přiřazování signálu výstupnímu portu v kódu. Tudíž byla analýza nadhodnocena a entita obsahuje moc vnořených podmínek a nestíhá pravděpodobně včas reagovat. Při použití více entit spojených sériově, aby program pracoval efektivněji, nastával problém s časováním, jelikož s použitím každé další entity se program zpožďuje o jednu periodu hodinového signálu.



Obr. 2.4: Simulace pro paket od pozice 0, který obsahuje IPv4 a UDP



Obr. 2.5: Simulace pro paket od pozice 0, který obsahuje IPv4 s UDP a IPv6 s UDP



Obr. 2.6: Simulace pro paket od pozice 192, který obsahuje IPv4 a UDP

2.1.2 Syntéza

Původně navržený kód sice optimálně fungoval v simulaci, ale již neprošel syntézou. Po provedení syntézy vyskočilo kritické varování. Pro jednu architekturu entity lze použít pouze jeden proces závislý na hodinovém signálu. Asynchronní logika není vhodná vzhledem k nepředvídatelnosti a možným hazardům. Původně plánovaný asynchronní reset by způsoboval zpomalení programu. Při syntéze moc velkých vstupně výstupních logických vektorů vyskočilo varování (logický vektor pro data 512 bit a pro součty 64 bit), že bylo použito moc vstupně výstupních prostředků. Takže velikosti logických vektorů ovlivňují počet použitých vstupně výstupních portů, jelikož zmenšení logických vektorů na půlku pomohlo k tomu, aby varování opět nenastalo. Avšak původně v simulaci navržené vyčítání bajtů z pole celková délka pro IPv4/IPv6 a počet bajtů z pole velikost v UDP protokolu, nakonec nepřipadá v úvahu. Vzhledem k tomu, že jedna maximální datová jednotka má 65535 bajtů, pro UDP a TCP se samozřejmě jedná o menší celkovou délku úseku. Alternativou by mohlo být čítat data po násobcích 32 bitů. Kvůli použití pouze malých logických vektorů a velké rychlosti zpracování dat programovatelného hradlového pole, nemůže být aplikována analýza prošlých paketů za sekundu.

V konečném návrhu byla zredukováno použití LUT na 129 oproti původním 1389 a 323 Flip-Flop oproti 933. Díky zmenšení vstupně výstupních vektorů byla zredukováno využití z 800 na 356. Bylo testováno na zařízení Zynq UltraScale+ ZCU106 Evaluation Platform (xczu7ev-ffvc1156-2-e). Z něhož pro LUT a Flip-Flop bylo využito 1 procento z celkově dostupných zdrojů, avšak pro vstupně výstupní prostředky 99 procent.

2.1.3 Testování

Otestování proběhlo na kartě Virtex 7 se síťovou kartou s propustností 100G k RAM, jejíž teoretický popis je uveden v teoretické části. Vstupní testovací pakety byly využity ze souboru typu .pcap, které se dají vygenerovat z programu Wireshark. Na jednom počítači byl spuštěný program Wireshark pro reálné zjištění síťového provozu a na druhém byla připojena síťová karta s programovatelným hradlovým polem a síťovou kartou s nahraným programem. Pomocí připojeného počítače ke kartě se testovala funkčnost a výsledky byly porovnány s výstupy z programu Wireshark. Praktické výsledky se shodovaly s výsledky ze simulace. Schéma pro testování je uvedeno viz 2.7. Při reálném testování sítě se dá karta připojit na jakýkoli uzel v síti, avšak pro sledování analýzy je potřeba i připojení počítače ke kartě. Na obrázku je příklad sledování provozu mezi dvěma hraničními směrovači 2.8.

Obr. 2.7: Schéma testování programu

Obr. 2.8: Schéma uplatnění v praxi

3 Závěr

Cílem bakalářské práce byl rozbor jazyka VHDL, seznámení se s hradlovými poli implementovanými se síťovou kartou na desce, vývojovým nástrojem NDK a prostředím Vivado. Jakožto nástroj analýzy sítě, byl navržen program pro čítání paketů ve VHDL. Konkrétně byly porovnány dvě karty typu NFB, popsána jejich teoretická funkčnost. Byl rozebrán model sítě a základní protokoly mezisíťové/síťové a transportní vrstvy s jejich záhlavími, informace byly dále využity pro sestavení programu. V práci jsou rozebrány paměti, což bylo klíčové pro pochopení funkcí na kartách NFB. V prostředí Xilinx Vivado byla provedena simulace a syntéza programu. Kdy bylo nutné vybalancovat mezi dokonalou funkčností programu a možností syntetizace do logických buněk a konečného zařízení. Byla provedena simulace a syntéza kódu navrženého v jazyce VHDL pro čítání paketů a základní vyzkoušení funkčnosti na kartě.

Literatura

[1] 4.1.2.1 The Physical Layer: INT_2018_st_1600, Chapter 4, Network acces. In: *Cisco: network academy* [online]. 2018 [cit. 2018-12-01]. Dostupné z: <https://static-course-assets.s3.amazonaws.com/ITN6/en/index.html#4.1.2.1>

[2] 05 Druhy optických vláken. BUBNÍK, Lukáš, Jiří KLAJBL a Petr MAZUCH. *Optoelektrotechnika*[online]. Code Creator, 2015 [cit. 2018-12-07]. ISBN 978-80-88058-20-5. Dostupné z: <https://publi.cz/books/185/05.html>

[3] DANĚK, Martin. Programovatelná hradlová pole — FPGA. *Automa: časopis pro automatizační techniku* [online]. 2002–, 2006(2) [cit. 26. 10. 2018]. Dostupné z URL: <http://automa.cz/cz/casopis-clanky/programovatelna-hradlova-pole-fpga-2006_02_30930_672>

[4] HOTSKÝ, Ondřej. *IPv6 a jeho praktické využití.* Praha, 2013. Diplomová práce. Bankovní institut vysoká škola Praha. Vedoucí práce Ing. Vladimír Beneš.

[5] KABELOVÁ, Alena a Libor DOSTÁLEK. *Velký průvodce protokoly TCP/IP a systémem DNS.* 5., aktualiz. vyd. Brno: Computer Press, 2008, 488 s. : il. ; 23 cm. ISBN 978-80-251-2236-5.

[6] KOLOUCH, Jaromír. Programovatelné logické obvody a hradlová pole – moderní stavební prvky číslicových systémů. *Automatizace: Číslicové obvody*[online].2009, **52**(1), 4 [cit. 2018-12-01]. Dostupné z: <http://www.urel.feec.vutbr.cz/web$_p$*ages/projekty/clanky/Kolouch*$_P$*LD.pdf* >

[7] KRÁL, Jiří. *Řešené příklady ve VHDL: Hradlová pole FPGA pro začátečníky.* Praha: BEN – technická literatura, 2010. 127 s. ISBN 9788073002572.

[8] MEET NFB-200G2QL, NEW 200G PROGRAMMABLE SMART NIC. *NETCOPE: TECHNOLOGIES*[online]. Brno: NETCOPE TECHNOLOGIES, 2018 [cit. 2018-12-01]. Dostupné z: <https://www.netcope.com/en/company/press-center/press-releases/meet-nfb-200g2ql,-new-200g-programmable-smart-nic>

[9] *NETCOPE TECHNOLOGIES*[online]. Brno: Netcope Technologies, 2018 [cit. 23. 11. 2018]. Dostupné z URL:<https://www.netcope.com/en>

[10] NETCOPE TECHNOLOGIES. *Netcope Development Kit: Firmware Developer's Manual.* Brno. Dostupné také z:<{url www.netcope.com>

[11] OREBAUGH, Angela. *Wireshark a Ethereal: kompletní průvodce analýzou a diagnostikou sítí*. Brno: Computer Press, 2008. 444 s. ISBN 978-80-251-2048-4.

[12] PETER, Winzer. Beyond 100G Ethernet. *Communications Magazine, IEEE*[online]. USA: IEEE, 2010, **48**(7) [cit. 2018-12-09]. DOI: 10.1109/M-COM.2010.5496875. ISSN 0163-6804

[13] PINKER, Jiří a Martin POUPA. *Číslicové systémy a jazyk VHDL*. 1. vydání. Praha: BEN – technická literatura, 2006. 349 s. ISBN 80-7300-198-5.

[14] POSTEL, J. *RFC 768*[online]. Information Sciences Institute, 1980. Dostupné také z: <`https://tools.ietf.org/html/rfc768`>

[15] PUŽMANOVÁ, Rita. *Moderní komunikační sítě od A do Z*. 2. aktualizované vydání. Brno: Computer Press, 2006. 430 s. ISBN 80-251-1278-0.

[16] SMEKAL, David, Jan HAJNY a Zdenek MARTINASEK. *Comparative Analysis of Different Implementations of Encryption Algorithms on FPGA Network Cards. IFAC PapersOnLine* [online]. Elsevier, 2018, textbf51(6), 312-317 [cit. 2018-12-05]. DOI: 10.1016/j.ifacol.2018.07.172. ISSN 2405-8963

[17] ŠŤASTNÝ, Jakub. *FPGA prakticky: realizace číslicových systémů pro programovatelná hradlová pole*. Praha: BEN - technická literatura, 2010. 200 s. ISBN 978-80-7300-261-9.

[18] *Xilinx*[online]. Xilinx, 2018 [cit. 2018-12-14]. Dostupné z:<`https://www.xilinx.com/`>

[19] WOLF, Wayne. *FPGA based system design*. New Jersey: Prentice-Hall, 2004. 530 s. ISBN 0-13-142461-0.

Seznam symbolů, veličin a zkratek

CLB	kombinační logické bloky z anglického combinational logic block
EOP	signál, který značí konec paketu z anglického End of packet
EOP_POS	signál, který značí místo konce paketu z anglického packet end position
Flip-flop	registr řízený hranou
FPGA	progrmovatelné hradlové pole anglického Field Programmable Gate Array
IP	protokol, který pracuje na síťové vrstvě anglického Internet Protocol
LUT	statická paměť, která využívá pravdivostní tabulku anglického lookup table
ISO	mezinárodní normalizační organice z anglického International Organisation for Standardization
OSA	otevřený systém architektury z anglického open systems architecture
OSI	referenční model z anglického Open System Interconnection
PDU	protokolová datová jednotka anglicky Protocol data unit
TCP	protokol, který pracuje na transportní vrstvě anglického Transmission Control Protocol
SAP	přístupové body služby z anglického Service Access Point
SOP	signál, který značí začátek paketu z anglického Start of packet
SOP_POS	signál, který značí místo začátku paketu z anglického packet start position
SRAM	statická paměť
UDP	protokol na transportní vrstvě z anglického User Datagram Protocol
VHDL	programovací jazyk složenina z anglických VHDL z VHSIC z Very High-Speed Integrated a VHDL z VHSIC Hardware Description Language